Unsupervised Deep learning for Next Generation sequencing Analysis

Thesis submitted in partial fulfillment of the requirement for the award of the degree of

Master of Engineering in Computer Science and Engineering

In the Faculty of Engineering and Technology

Jadavpur University

By

Biswajit Das

Exam Roll No **:** M4CSE22033

Registration No **: 154157** of **2020-2021**

Under the Guidance of

Assistant Professor (Dr.) Anasua Sarkar

Department of Computer Science and Engineering

Jadavpur University

Kolkata - 700 032

2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE

I hereby recommend that the thesis entitled **"Unsupervised Deep learning for Next Generation sequencing Analysis"** prepared under my supervision by **Biswajit Das,** Exam- Roll No**: M4CSE22033**, Registration No: **154157 of 2020-2021** be accepted in partial fulfillment of the requirements for the degree of **Master of Engineering in Computer Science and Engineering** of **Jadavpur University, Kolkata**.

Supervisor

Asst Prof. (Dr.) Anasua Sarkar

Department of Computer Science & Engineering,

Jadavpur University, Kolkata - 32

Dr. Nandini Mukhopadhyay

Head of the Department (HOD)

Department of Computer Science & Engineering

Jadavpur University, Kolkata – 32

Dr. Chandan Mazumdar

DEAN

Faculty Council of

Engineering and Technology

JADAVPUR UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

APPROVAL FROM EXAMINERS

The foregoing thesis "**Unsupervised Deep learning for Next Generation sequencing Analysis**" at instance is hereby approved as a creditable study of an engineering subject carried out and presented in a manner of satisfactory to warrant its acceptance as pre-requisite to the degree for which it has been submitted. It is notified to be understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed and conclusion drawn there in but approve the thesis only for the purpose for which it has been submitted.

Final Examination for the Evaluation of Thesis

Board of Examiners

_____

_____

(Signature of Examiners)

ACKNOWLEDGEMENTS

I express my profound gratitude and sincere thanks to **Dr. Anasua Sarkar Assistant Professor, Jadavpur University** for her valuable suggestions, guidance, constant encouragement and intent supervision at every stage of my thesis work. It has been a great learning process for me. It is in his association that gave me opportunities to enhance my skills and knowledge which I did not even know earlier.

Last but not the least, I express my gratitude to my friends and family for their constant support and unfailing guidance in whatever I did.

Date :

Place : Kolkata

Biswajit Das

M.E. (C.S.E)

Roll No: 002010502032

Contents

Abstract

In this thesis, we attempt to solve the problem of Next Generation RNA-sequencing analysis. **This analysis aims to compare the expression levels of multiple genes between two or more samples, under specific circumstances or in a specific cell to give a global picture of cellular function. Thanks to these advances, gene expression data are being generated in large throughput. A Deep Learning approach and recognition of Next Generation RNA-sequencing analysis.** One of the primary data analysis tasks for gene expression studies involves data-mining techniques such as clustering and classification. RNA-sequencing technologies, which sequence the RNA molecules being transcribed in cells, allow exploration of the process of transcription in exquisite detail. Challenges, critical parameters, and possible downstream functional analysis pipelines associated with each step are highlighted and discussed. This provides a comprehensive understanding of state-of-the-art RNA-seq analysis pipeline and a greater understanding of the transcriptase.

Chapter 1

Introduction

Due to advancement in technology and drop in cost for high-throughput profiling of molecular assay and next-generation sequencing, RNA sequencing (RNA-seq) has becoming a common tool for scientists to study the transcriptomic phenomenon observed in biological samples. RNA-seq data allows one to study the system-wide transcriptional changes from a variety of aspects, ranging from expression changes in gene or isoform levels, to complex analysis like discovery of novel, alternative or cryptic splicing sites, RNA-seq read alignment is further complicated by the presence of processed pseudo genes in the reference genome. Pseudo genes often have highly similar sequences to functional, intron-containing genes. In most cases, the pseudo gene versions are not transcribed [1], although this suggestion has recently been disputed [2].

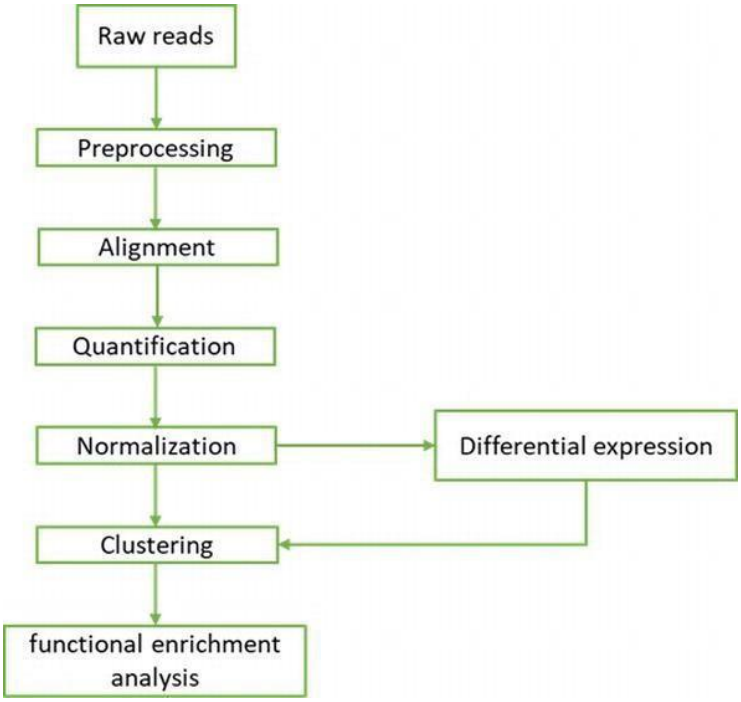
1.1 The Objective

The primary objective of this chapter is to present algorithms for clustering gene expression data from RNA-seq. Therefore, in the first section, we will describe the different steps of the gene expression analysis workflow from preprocessing the raw reads to gene expression clustering and classification. In the second part of the chapter we will describe traditional, model-based and machine learning clustering methods for gene expression data, then we will conclude this chapter with a study for clustering samples of four public datasets from recount2, using different clustering methods and also evaluating the performance of each one using the adjusted rand index and accuracy. In our analysis of RNA-seq reads from multiple human samples [3, 4], genes with processed pseudo genes seem to be expressed at higher levels compared with other genes (see Results and discussion). Although this observation has not been explored thoroughly, a plausible explanation is that genes with higher levels of expression may, over the

course of evolution, have had an increased chance of being picked up by transposons and re-integrated into the genome, creating pseudo gene copies.

1.2 DIFFERENTIAL GENE EXPRESSION ANALYSIS OF RNA-SEQ

The immediate question one could ask from an experiment with RNA-seq is what genes are deregulated due to the designed perturbation, treatment, etc. Thus, the first protocol consists of the basic pipeline for analysing raw sequence reads of RNA data to reveal the set of significantly deregulated genes. Specifically, this pipeline consists of five main steps, where each step corresponds to one phase of the analysis that achieve certain milestone. A typical RNA-seq data analysis workflow starts by pre-processing raw reads for contamination removal and quality control checks. The following step is to align the reads to a reference genome, or to make a de novo assembly if there is not any. Following the alignment, the quantification step aims to quantify aligned reads to produce a count matrix to use as entry data for Differential Expression (DE) analysis. We will perform Normalization of the raw counts separately and do the clustering without going through differential gene expression analysis. In the following section we describe with more details each step of the pipeline (Figure 1)



(Figure -1)

Chapter 2

Processing

Preprocessing raw reads consist of checking the quality of the reads, adapters trimming, removal of short reads and filtering bad quality bases. Tools like FastQC can generate a report summarizing the overall quality of the sequence information [5]. Based on this report we can determine how the quality trimming should be set up. Trimmomatic is one of many tools used to clean up the raw data. It can be used to remove adapters from the reads, trim off any low-quality bases at the ends of reads, and filter short reads that can align to multiple locations on the reference genome. Once the trimming step is done, it is a good practice to recheck the quality of the reads by rerunning FastQC. RNA-seq mapping algorithms have two additional challenges. First, because genes in eukaryotic genomes contain introns, and because reads sequenced from mature mRNA transcripts do not include these introns, any RNA-seq alignment program must be able to handle gapped (or spliced) alignment with very large gaps.

Alignment

More important for the alignment problem is that around 20% of junction-spanning reads extend by 10 bp or less into one of the exons they span. These small 'anchors' make it extremely difficult for alignment software to map reads accurately, particularly if the algorithm relies (as most do) on an initial mapping of fixed-length k-mers to the genome. This initial mapping, using exact matches of k-mers, is crucial for narrowing down the search space into small local regions in which a read is likely to align. If a read extends only a few bases into one of two adjacent exons, then it often happens that the read will align equally well, but incorrectly, with the sequence of the intervening intron. In RNA-seq, alignment is a major step for the calculation of transcript or gene expression levels; several splice-aware alignment methods have been developed for RNA-seq experiments such as STAR, HISAT2 or Top Hat. These aligners are designed to specifically address many of the challenges of RNA-seq data mapping using a strategy to account for spliced alignment

Quantification

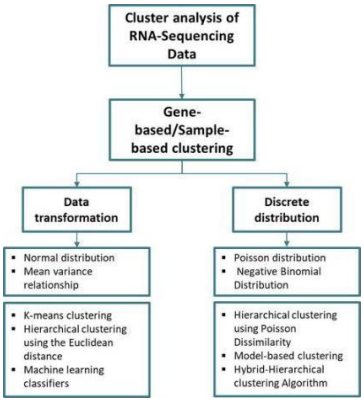
Quantification of gene expression is to count the number of reads that map to each gene using methods such as HTSeq-count, Feature Counts or kallisto [6, 7, 8]. This step is crucial if we want to do a gene differential expression analysis, which means to identify genes (or transcripts), if any, that have a statistically significant difference in abundance across the experimental groups or conditions.

Normalization

The read counts generated in the quantification step need to be normalized to make accurate comparisons of gene expression between samples or when doing an exploratory data analysis. Several normalization methods are used for this purpose. Although this observation has not been explored thoroughly, a plausible explanation is that genes with higher levels of expression may, over the course of evolution, have had an increased chance of being picked up by transposons and re-integrated into the genome, creating pseudo gene copies.

Clustering

Cluster analysis techniques have proven to be helpful to understand gene expression data by uncovering unknown relationships among genes and unveiling different subtypes of diseases when it comes to clustering biological samples. Clustering transcriptomes profiled by scRNA-seq has been routinely conducted to reveal cell heterogeneity and diversity. However, clustering analysis of scRNA-seq data remains a statistical and computational challenge, due to the pervasive dropout events obscuring the data matrix with prevailing 'false' zero count observations. Here, we have developed scDeepCluster, a single-cell model-based deep embedded clustering method, which simultaneously learns feature representation and clustering via explicit modelling of scRNA-seq data generation.



Chapter 3

Clustering With Deep Learning

Deep learning is also a technique that can be used to learn better data representation of high-dimensional data. The two recently published surveys [9, 10] present a taxonomy of existing deep clustering algorithms, by describing the different Neural Network Architecture that exists for feature representation, clustering loss function and Performance Evaluation Metrics for Deep Clustering. In [11], the authors categorize current deep clustering models into following three categories: **One method to do deep learning based clustering is to learn good feature representations and then run any classical clustering algorithm on the learned representations**. There are several deep unsupervised learning methods available which can map data-points to meaningful low dimensional representation vectors. One popular method to learn meaningful representations is deep auto-encoders. Here the input is fed into a multilayer encoder which has a low dimensional output. That output is fed to a decoder which produces an output of the same size as input. The training objective of the model is to reconstruct the given input.

3.1 Working principles of DL-based clustering methods

Let us consider the problem of clustering of nn samples, XX = {x(1),x(2),...,x(n)}{x(1),x(2),...,x(n)} into KK-categories, each represented by a centroid μj,j=1,…,Kμj,j=1,…,K where X∈RDX∈RD. In pipeline methods, more or less a similar working principle is followed in which a DL-based clustering algorithm is usually trained in two phases:

·**Phase 1:** parameter initialization and RL with a DNN architecture and training using non-clustering *loss (e.g. standard RL1).* Then clustering-friendly representations of the data called latent features (LFs) are extracted from one or more layers (depending on the type of network architecture).

Phase 2: parameter optimization by iterating between computing an auxiliary target distribution and minimizing *clustering* loss [e.g. Kullback–Leibler divergence (KLD) [12] and cluster assignment hardening loss (CAHL)] in which cluster assignments are formulated, followed by the centroid updated with the back propagation in which an ML-based clustering algorithm is applied to optimize the clustering objective iteratively. In particular, AC [13] and K-means [14, 15, 16] algorithms are broadly used in the literature.

Followed by this principle, instead of clustering the samples directly in the original input space XX, it is transformed with a nonlinear mapping $fθ:X→Zfθ:X→Z$ where $θθ$ are learnable parameters and $Z∈RKZ∈RK$ is the learned or embedded feature space, where $K≪DK≪D$. To parameterize $fθfθ$, a DNN architecture such as AEs is used due to their function approximation properties and feature learning capabilities However, for a better clustering result, the network is often trained and updated to optimize both clustering and non-clustering losses jointly in phase 2. Concisely, the following three steps are broadly involved in existing approaches:

Unsupervised Learning:

Unsupervised leaning is different from the supervised learning technique. The machine is trained using the unlabeled dataset and the predicts the output without any supervision. Algorithms like K-means clustering [17], etc. are some of the common unsupervised learning algorithms.

Neural Networks:

Artificial neural networks are a popular type of supervised learning Model. A special case of a neural network called the convolutional Neural network (CNN) is the primary focus of this thesis. The name ‘Artificial Neural Networks’ was given to this model because they were developed to imitate the neural function of the human brain. An artificial neural network consists of a set of neurons connected to each other and are grouped into layers to replicate the neural function of out brain. Similar to the neurons in a human brain, the neurons in an artificial neural network function as units of calculation. The connections between neurons are known as ‘synapses’ which are nothing but weighted values. Therefore, in a simple sense, when an input value is provided at a neuron (x1, x2, ...), it traverses the synapse,

multiplying its value with the weighted value of the synapse (w1, w2, ...) as shown in the Figure 2.2. Bias 'b' is then added to the summation of these values. This will be the output of the neuron. Since a neuron does not know its boundary, a mapping mechanism is required to map the inputs to the output, known as the 'Activation function' [21]. In a fully connected feed- forward multi-layer network, all the outputs of a layer of a layer of neurons is fed as input to every neuron of the next layer. As a result, some of layers get to process the original input data, while some layers get to process the data that has been obtained from neurons from the previous layer. Therefore, the number of weights of any neuron in the network is equal to the number of neurons in the layer previous to the layer of the neuron [19].

$$Y = \sum_{k=1}^{n} (wn * xn) + b \qquad \text{.....} \quad (1.1)$$

In the above equation, 'x' is the input value given at the neuron, 'w' is the weighted value of the synapse, 'n' is the number of neurons, 'b' is the bias and 'y' is the output of the network. Therefore, according to the equation (1.1), the value of output 'y' is equal to the summation of the product corresponding weights and bias 'b'.A multi-layered artificial neural network, as shown above includes three types of layers: an input layer, one or more hidden layers and an output layer [19]. The input layer usually merely passes data along without modifying it. Most of the computation happens in the hidden layers. The output layer converts the hidden layer activation to an output, such as a classification. The outputs of each hidden layer serve as the inputs for the next hidden layer. The number of neurons in the output layer is equal to the number of classes trained for the neural network. Deep learning helped solve many problems that were historically challenging for classical software development tools and approaches. For instance, years ago, machine learning engineers were able to create a software that could predict breast cancer survival windows better than human experts. However, building the features of the software required the efforts of dozens of engineers and breast cancer experts and took a lot of time develop. Deep learning provides a fundamentally different approach to doing machine learning. Deep learning relies on neural networks, a general-purpose function that can solve any problem representable through examples. When we provide a neural network with many labeled examples of a specific kind of data, it will be able to extract common

Chapter 4

Deep learning Frameworks

DL approaches for analyzing NGS data, such as Deep Bind, had to be built from scratch, which can be challenging. Fortunately, excellent frameworks for the implementation of deep networks now exist that greatly facilitate the network development. Such frameworks provide common activation functions, handle gradient computation, training, and usually feature optimized implementations for different accelerator architectures, such as GPUs or tensor processing units (TPUs). Recently, large Cloud providers have also started to offer DL platforms. This includes facilities for distributed computation of TF- or PyTorch-jobs, but also offerings such as Google's Cloud AutoML or Amazon's Sage Maker, which promise mostly automated model training. In these packages, end-users can typically, using an intuitive graphical user interface, upload the training data, choose what kind of ML to perform, and highlight the property that is to be inferred. The framework will then automatically build an appropriate ML model and train it on the provided data. With the growing popularity of DL methods for analyzing sequencing data, several software frameworks and packages specifically designed for bioinformatics data have been introduced recently. Libraries, such as Nucleus or Janggu , can be used alongside Keras, TF, or PyTorch. They offer dedicated objects for processing biological sequence data, which makes it easy to read, write, analyze, and visualize data in common genomics file formats, such as BAM, FASTA, bigWig, VCF, or BED.

Chapter 5

Clustering with unsupervised representation learning

One method to do deep learning based clustering is to learn good feature representations and then run any classical clustering algorithm on the learned representations. There are several deep unsupervised learning methods available which can map data-points to meaningful low dimensional representation vectors. The representation vector contains all the important information of the given data-point, hence clustering on the representation vectors yield better results. One popular method to learn meaningful representations is deep auto-encoders. Here the input is fed into a multilayer encoder which has a low dimensional output. That output is fed to a decoder which produces an output of the same size as input.

Clustering via information maximization:

Regularized Information Maximization is an information theoretic approach to perform clustering which takes care of class separation, class balance, and classifier complexity. The method uses a differentiable loss function which can be used to train multi-logit regression models via back propagation. The training objective is to maximize the mutual information between the input *x* and the model output *y* while imposing some regularization penalty on the model parameters. Mutual information can be represented as the difference between marginal entropy and conditional entropy. Hence the training objective to minimize is:

$$\mathcal{R}_{\mathrm{SAT}}(\theta; T) - \lambda \left[H(Y) - H(Y|X) \right]$$

Here it is maximizing the marginal entropy *H(Y)* and minimizing the conditional entropy *H(Y|X)*.

By maximizing *H(Y)*, the cluster assignments are diverse, hence the model cannot degenerate by assigning a single cluster to all the input data points. In fact, it will try to make the distribution of clusters as uniform as possible because entropy will be maximum when the probability of each cluster is the same.

The neural network model with the softmax activation estimates the conditional probability $p\ (\ y|x\ )$. By minimizing $H\ (\ Y|X\ )$, it ensures that the cluster assignment of any data point is with high confidence. If $H(\ Y|X\ )$ is not minimized and only $H(Y)$ is maximized, the model can degenerate by assigning an equal conditional probability to each cluster given any input.

While implementing in order to compute H(Y), p(y) is computed by marginalizing $p(\ y|x\ )$ over a mini-batch. For a given x, $p(\ y|x\ )$ is the output of the network after the softmax activation.

Chapter 6

Experiment & Results

In this chapter, all the related experiment and its results are given. We will see how the datasets are collected.

ProbeNam	p (Corr) (	p ([C] vs [	p ([D] vs [	p ([DT] vs	p ([DT] vs	FC ([C] vs	Log FC ([C	FC (abs)	Regulatio
A_55_P28	0.046765	0.013584	0.013642		0.049415	-11.089	-3.47105	11.08898	down
A_55_P24	0.033984	0.007776	0.010061		0.017742	-12.0979	-3.59668	12.09789	down
A_66_P12	0.014925	0.001923	0.002342		0.006321	-23.3078	-4.54274	23.30775	down
A_55_P28	0.003077	0.000159	0.000468		0.00053	-32.6124	-5.02735	32.61242	down
A_55_P21	0.019382	0.002959	0.007758		0.003831	-9.43362	-3.23781	9.43362	down
A_52_P11	0.002123	0.000087	0.000397		0.000379	-17.3665	-4.11823	17.36647	down
A_66_P13	0.009354	0.000881	0.001083		0.004077	5.511046	2.462326	5.511046	up

There are several similarity measures for cluster evaluation, we chose to work with the adjusted Rand index which is the corrected-for-chance version of the Rand index. It is a measure used in data clustering to evaluate the performance of a clustering method, by comparing the results of a clustering algorithm against known classes from external criteria [20]. In our study, we performed different sample-based classification method on four different datasets, after that, we compared the results to the class labels we associated to each sample based on the field "characterization of the samples" in the phenotype table in recount2, and then we used the ARI for cluster validation. Instead of trying to precisely mimic real RNA-seq experiments, which may not be possible in any practical sense, we generated data with relatively simple settings and expression levels, calculated using a model from the Flux Simulator system [21], as follows. For the first test set, we generated reads from the known transcripts on the entire human genome without introducing any mismatches we then generated additional datasets, in which we included 1) insertions and deletions into the known transcripts at random locations, and 2) insertions and deletions in the reads themselves to mimic sequencing errors

6.2 Training Process

6.2.1. Experimental Environment

Hardware environment: Intel(R)-Core (TM) I3-5790 2.70GHz CPU

Software environment: Windows 10, shyphy,

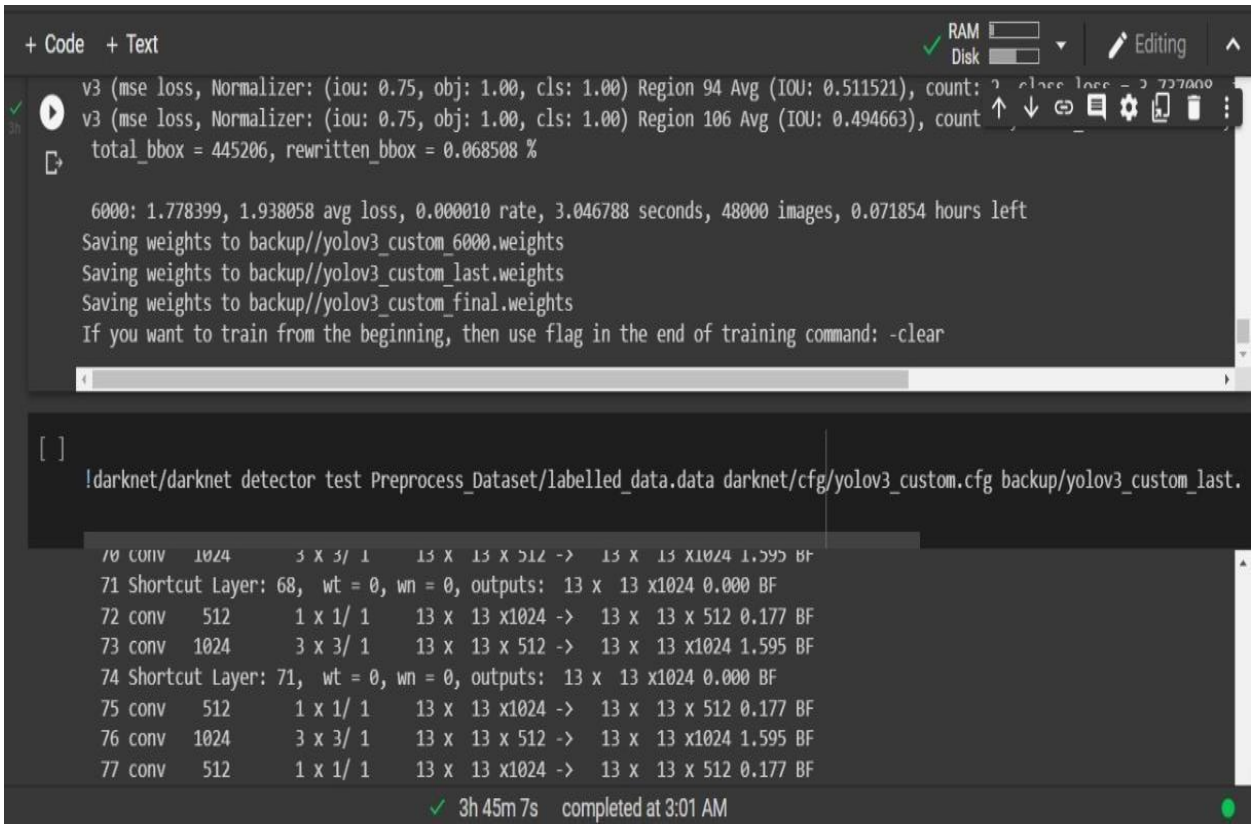
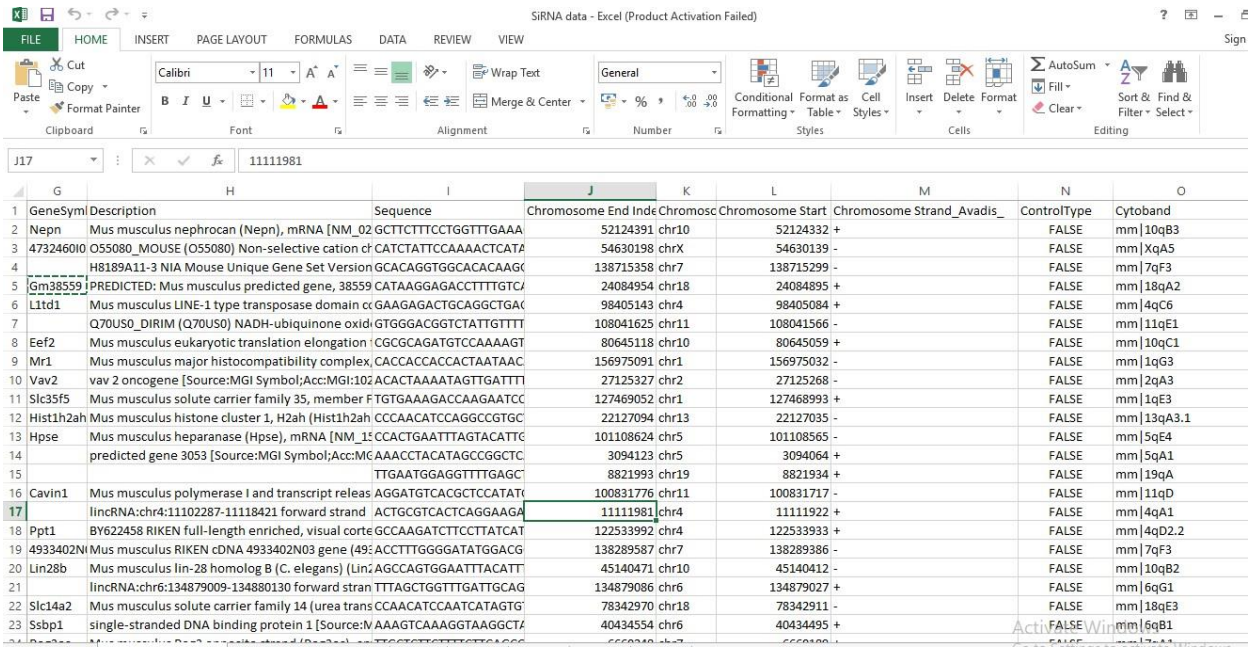
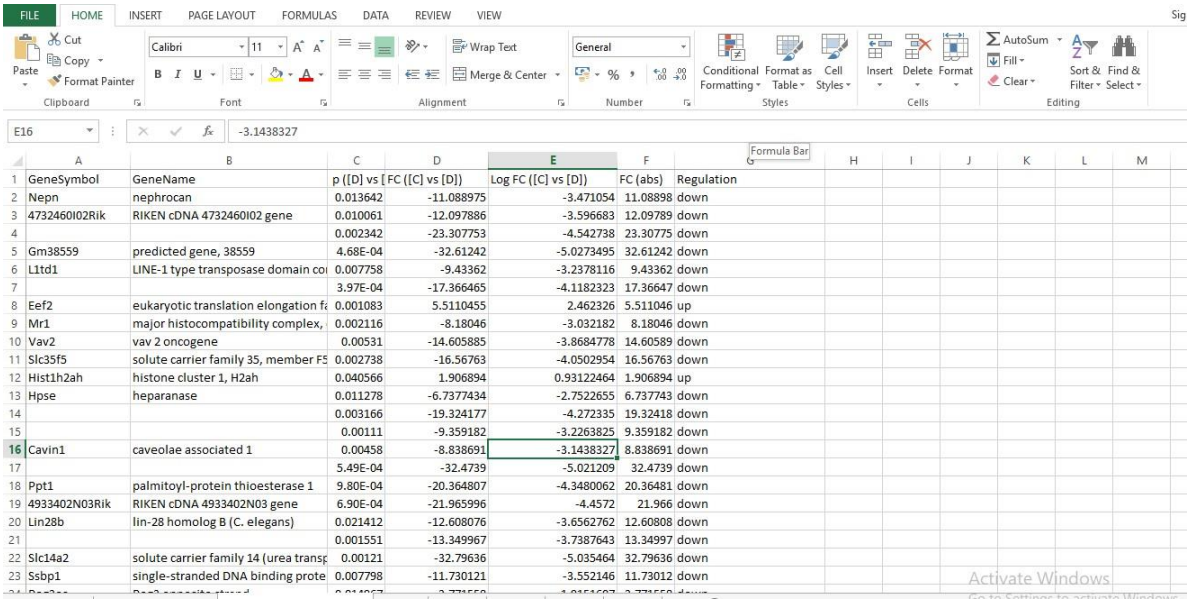


Fig : During Training Process

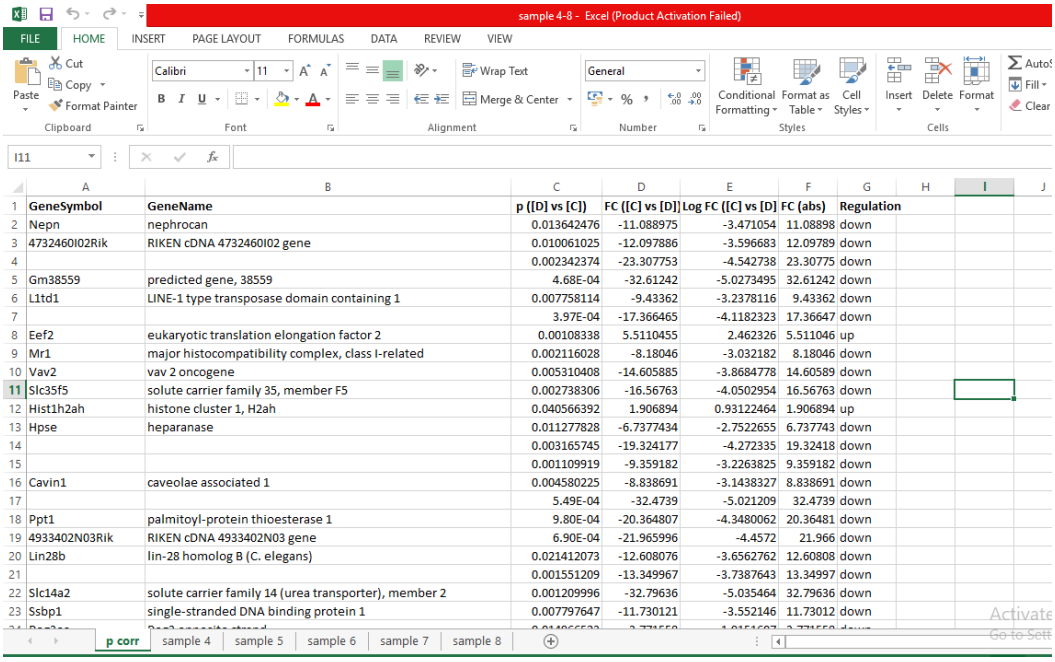


	G	H	I	J	K	L	M	N	O
1	GeneSymbol	Description	Sequence	Chromosome End Inde	Chromosc	Chromosome Start	Chromosome Strand_Avadis_	ControlType	Cytoband
2	Nepn	Mus musculus nephrocan (Nepn), mRNA [NM_02	GCTTCTTTCCTGGTTTGAAA	52124391	chr10	52124332	+	FALSE	mm 10qB3
3	4732460I0	O55080_MOUSE (O55080) Non-selective cation ch	CATCTATTCCAAAACTCATA	54630198	chrX	54630139	-	FALSE	mm XqA5
4		H8189A11-3 NIA Mouse Unique Gene Set Version	GCACAGGTGGCACACAAG	138715358	chr7	138715299	-	FALSE	mm 7qF3
5	Gm38559	PREDICTED: Mus musculus predicted gene, 38559	CATAAGGAGACCTTTTGTC	24084954	chr18	24084895	+	FALSE	mm 18qA2
6	L1td1	Mus musculus LINE-1 type transposase domain co	GAAGAGACTGCAGGCTGA	98405143	chr4	98405084	+	FALSE	mm 4qC6
7		Q70US0_DIRIM (Q70US0) NADH-ubiquinone oxido	GTGGGACGGTCTATTGTTTT	108041625	chr11	108041566	-	FALSE	mm 11qE1
8	Eef2	Mus musculus eukaryotic translation elongation	CGCGCAGATGTCCAAAAGT	80645118	chr10	80645059	+	FALSE	mm 10qC1
9	Mr1	Mus musculus major histocompatibility complex,	CACCACCACCACTAATAAC	156975091	chr1	156975032	-	FALSE	mm 1qG3
10	Vav2	vav 2 oncogene [Source:MGI Symbol;Acc:MGI:102	ACACTAAAATAGTTGATTTT	27125327	chr2	27125268	-	FALSE	mm 2qA3
11	Slc35f5	Mus musculus solute carrier family 35, member F	TGTGAAAGACCAAGAATCC	127469052	chr1	127468993	+	FALSE	mm 1qE3
12	Hist1h2ah	Mus musculus histone cluster 1, H2ah (Hist1h2ah	CCCAACATCCAGGCCGTGC	22127094	chr13	22127035	-	FALSE	mm 13qA3.1
13	Hpse	Mus musculus heparanase (Hpse), mRNA [NM_15	CCACTGAATTTAGTACATTC	101108624	chr5	101108565	-	FALSE	mm 5qE4
14		predicted gene 3053 [Source:MGI Symbol;Acc:MG	AAACCTACATAGCCGGCTC	3094123	chr5	3094064	+	FALSE	mm 5qA1
15			TTGAATGGAGGTTTTGAGC	8821993	chr19	8821934	+	FALSE	mm 19qA
16	Cavin1	Mus musculus polymerase I and transcript releas	AGGATGTCACGCTCCATAT	100831776	chr11	100831717	-	FALSE	mm 11qD
17		lincRNA:chr4:11102287-11118421 forward strand	ACTGCGTCACTCAGGAAGA	11111981	chr4	11111922	+	FALSE	mm 4qA1
18	Ppt1	BY622458 RIKEN full-length enriched, visual corte	GCCAAGATCTTCCTTATCAT	122533992	chr4	122533933	+	FALSE	mm 4qD2.2
19	4933402N	Mus musculus RIKEN cDNA 4933402N03 gene (49	ACCTTTGGGGATATGGACG	138289587	chr7	138289386	-	FALSE	mm 7qF3
20	Lin28b	Mus musculus lin-28 homolog B (C. elegans) (Lin2	AGCCAGTGGAATTTACATT	45140471	chr10	45140412	-	FALSE	mm 10qB2
21		lincRNA:chr6:134879009-134880130 forward stran	TTTAGCTGGTTTGATTGCAG	134879086	chr6	134879027	+	FALSE	mm 6qG1
22	Slc14a2	Mus musculus solute carrier family 14 (urea trans	CCAACATCCAATCATAGTG	78342970	chr18	78342911	-	FALSE	mm 18qE3
23	Ssbp1	single-stranded DNA binding protein 1 [Source:M	AAAGTCAAAGGTAAGGCTA	40434554	chr6	40434495	+	FALSE	mm 6qB1



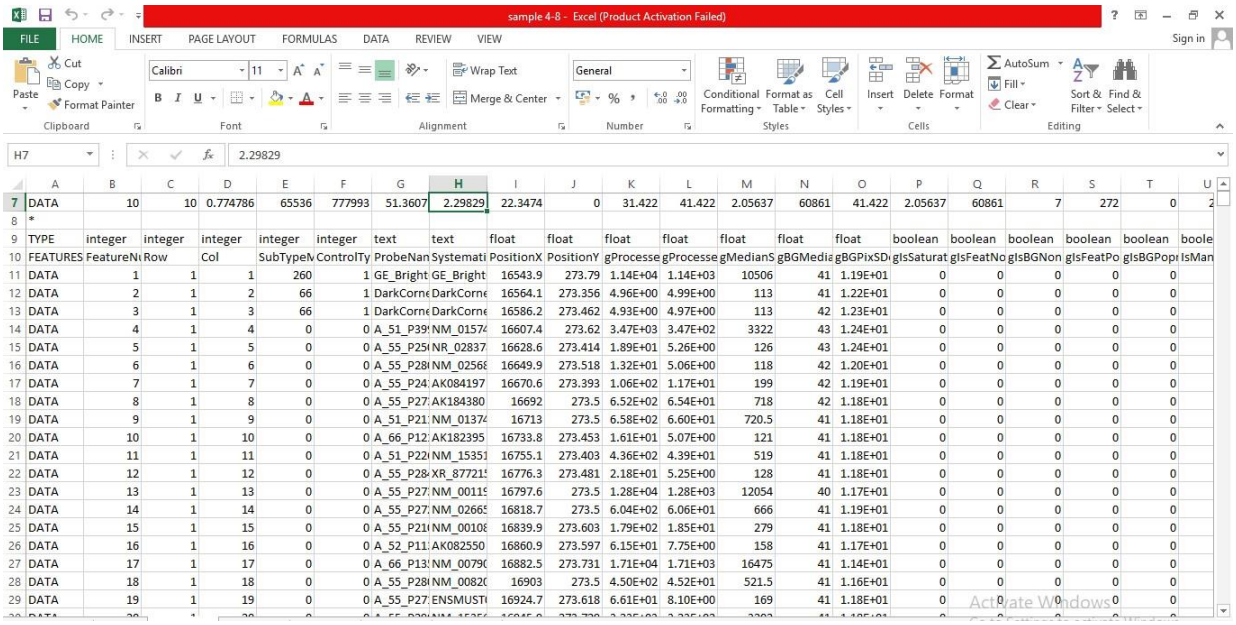
	A	B	C	D	E	F	G	H	I	J	K	L	M
1	GeneSymbol	GeneName	p ([D] vs [	FC ([C] vs [D])	Log FC ([C] vs [D])	FC (abs)	Regulation						
2	Nepn	nephrocan	0.013642	-11.088975	-3.471054	11.08898	down						
3	4732460I02Rik	RIKEN cDNA 4732460I02 gene	0.010061	-12.097886	-3.596683	12.09789	down						
4			0.002342	-23.307753	-4.542738	23.30775	down						
5	Gm38559	predicted gene, 38559	4.68E-04	-32.61242	-5.0273495	32.61242	down						
6	L1td1	LINE-1 type transposase domain co	0.007758	-9.43362	-3.2378116	9.43362	down						
7			3.97E-04	-17.366465	-4.1182323	17.36647	down						
8	Eef2	eukaryotic translation elongation fa	0.001083	5.5110455	2.462326	5.511046	up						
9	Mr1	major histocompatibility complex,	0.002116	-8.18046	-3.032182	8.18046	down						
10	Vav2	vav 2 oncogene	0.00531	-14.605885	-3.8684778	14.60589	down						
11	Slc35f5	solute carrier family 35, member F5	0.002738	-16.56763	-4.0502954	16.56763	down						
12	Hist1h2ah	histone cluster 1, H2ah	0.040566	1.906894	0.93122464	1.906894	up						
13	Hpse	heparanase	0.011278	-6.7377434	-2.7522655	6.737743	down						
14			0.003166	-19.324177	-4.272335	19.32418	down						
15			0.00111	-9.359182	-3.2263825	9.359182	down						
16	Cavin1	caveolae associated 1	0.00458	-8.838691	-3.1438327	8.838691	down						
17			5.49E-04	-32.4739	-5.021209	32.4739	down						
18	Ppt1	palmitoyl-protein thioesterase 1	9.80E-04	-20.364807	-4.3480062	20.36481	down						
19	4933402N03Rik	RIKEN cDNA 4933402N03 gene	6.90E-04	-21.965996	-4.4572	21.966	down						
20	Lin28b	lin-28 homolog B (C. elegans)	0.021412	-12.608076	-3.6562762	12.60808	down						
21			0.001551	-13.349967	-3.7387643	13.34997	down						
22	Slc14a2	solute carrier family 14 (urea transp	0.00121	-32.79636	-5.035464	32.79636	down						
23	Ssbp1	single-stranded DNA binding prote	0.007798	-11.730121	-3.552146	11.73012	down						

Sample: 2



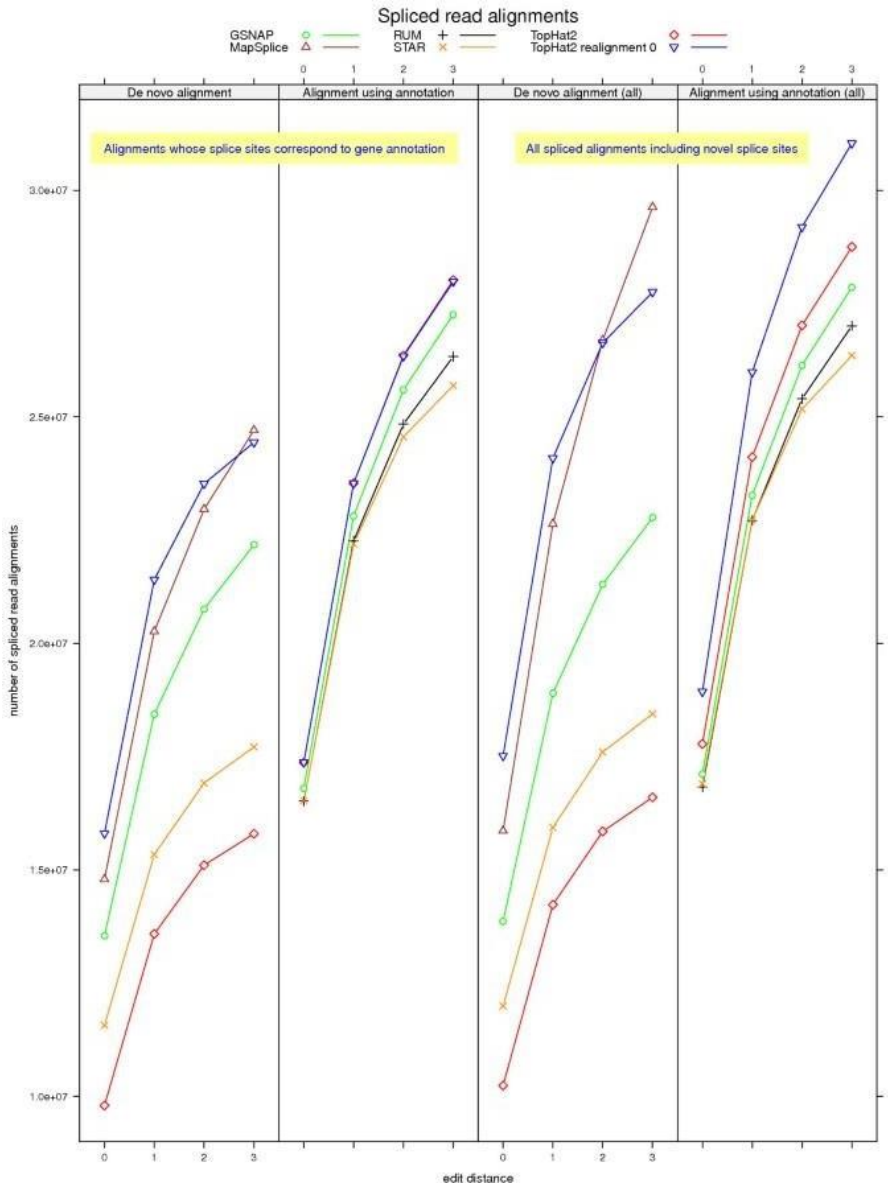
sample 4-8 - Excel (Product Activation Failed)

	GeneSymbol	GeneName	p ([D] vs [C])	FC ([C] vs [D])	Log FC ([C] vs [D]	FC (abs)	Regulation
2	Nepn	nephrocan	0.013642476	-11.088975	-3.471054	11.08898	down
3	4732460I02Rik	RIKEN cDNA 4732460I02 gene	0.010061025	-12.097886	-3.596683	12.09789	down
4			0.002342374	-23.307753	-4.542738	23.30775	down
5	Gm38559	predicted gene, 38559	4.68E-04	-32.61242	-5.0273495	32.61242	down
6	L1td1	LINE-1 type transposase domain containing 1	0.007758114	-9.43362	-3.2378116	9.43362	down
7			3.97E-04	-17.366465	-4.1182323	17.36647	down
8	Eef2	eukaryotic translation elongation factor 2	0.00108338	5.5110455	2.462326	5.511046	up
9	Mr1	major histocompatibility complex, class I-related	0.002116028	-8.18046	-3.032182	8.18046	down
10	Vav2	vav 2 oncogene	0.005310408	-14.605885	-3.8684778	14.60589	down
11	Slc35f5	solute carrier family 35, member F5	0.002738306	-16.56763	-4.0502954	16.56763	down
12	Hist1h2ah	histone cluster 1, H2ah	0.040566392	1.906894	0.93122464	1.906894	up
13	Hpse	heparanase	0.011277828	-6.7377434	-2.7522655	6.737743	down
14			0.003165745	-19.324177	-4.272335	19.32418	down
15			0.001109919	-9.359182	-3.2263825	9.359182	down
16	Cavin1	caveolae associated 1	0.004580225	-8.838691	-3.1438327	8.838691	down
17			5.49E-04	-32.4739	-5.021209	32.4739	down
18	Ppt1	palmitoyl-protein thioesterase 1	9.80E-04	-20.364807	-4.3480062	20.36481	down
19	4933402N03Rik	RIKEN cDNA 4933402N03 gene	6.90E-04	-21.965996	-4.4572	21.966	down
20	Lin28b	lin-28 homolog B (C. elegans)	0.021412073	-12.608076	-3.6562762	12.60808	down
21			0.001551209	-13.349967	-3.7387643	13.34997	down
22	Slc14a2	solute carrier family 14 (urea transporter), member 2	0.001209996	-32.79636	-5.035464	32.79636	down
23	Ssbp1	single-stranded DNA binding protein 1	0.007797647	-11.730121	-3.552146	11.73012	down

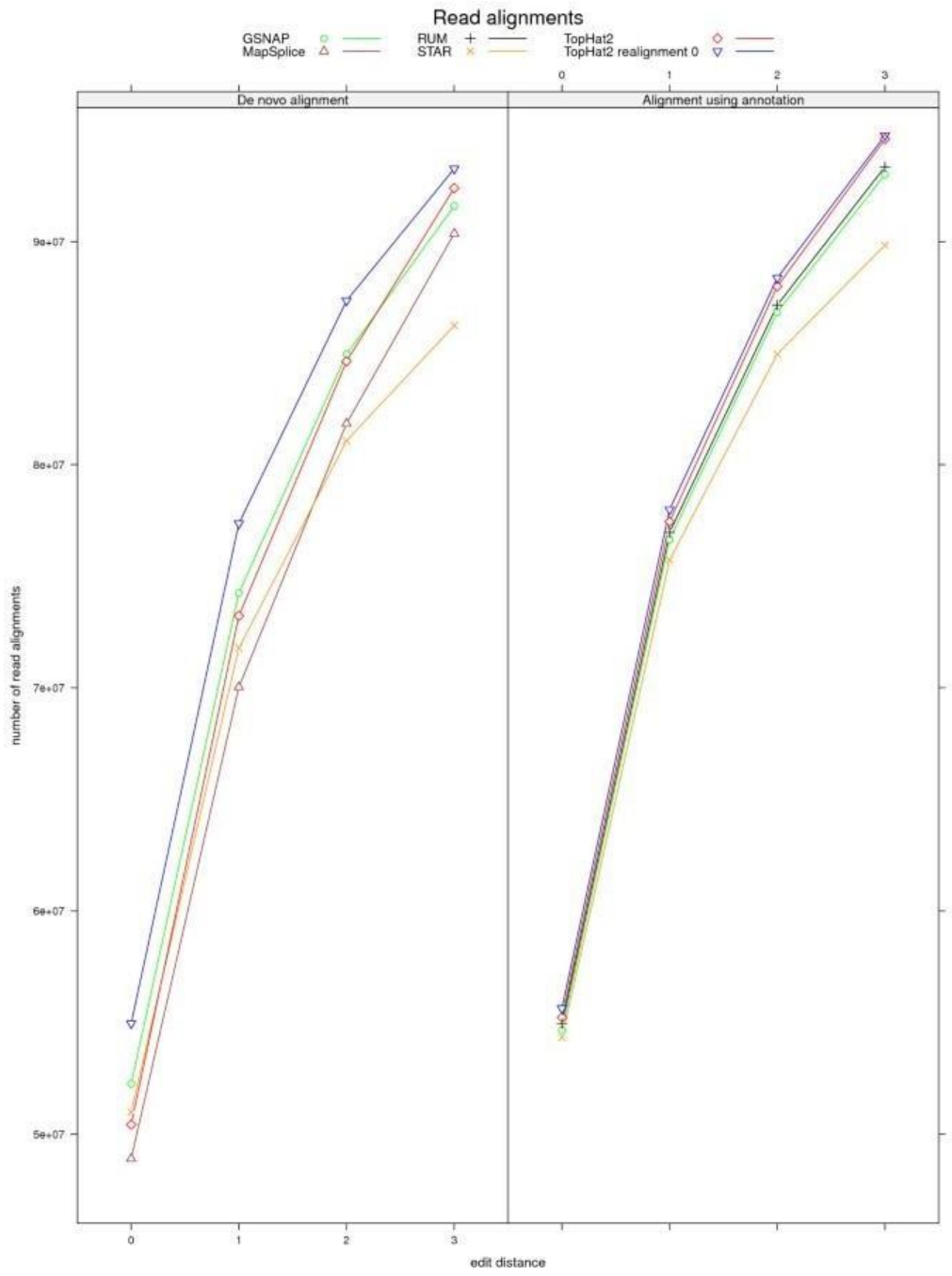


sample 4-8 - Excel (Product Activation Failed)

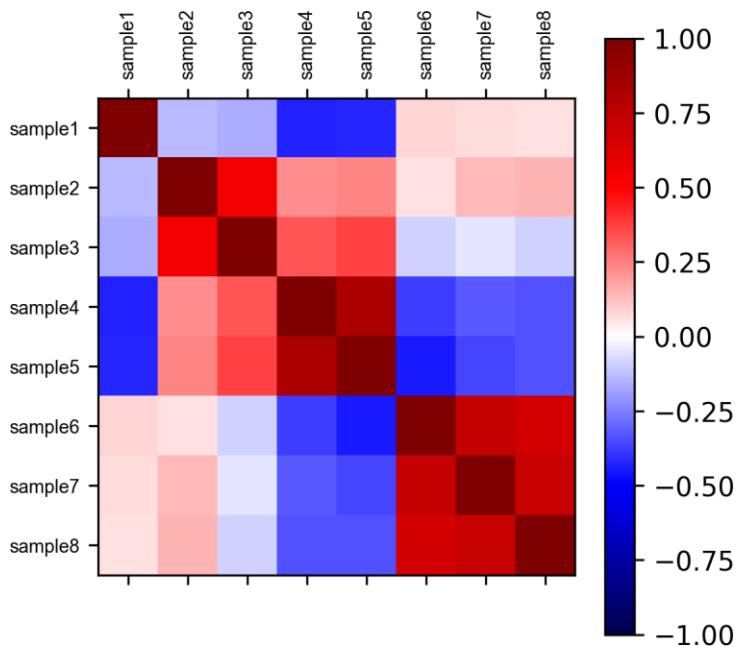
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
7	DATA	10	10	0.774786	65536	777993	51.3607	2.29829	22.3474	0	31.422	41.422	2.05637	60861	41.422	2.05637	60861	7	272	0	2
8	*																				
9	TYPE	integer	integer	integer	integer	integer	text	text	float	float	float	float	float	float	float	boolean	boolean	boolean	boolean	boolean	boole
10	FEATURES	FeatureNu	Row	Col	SubTypeN	ControlTy	ProbeNam	Systemati	PositionX	PositionY	gProcesse	gProcesse	gMedianS	gBGMedia	gBGPixSD	gIsSaturat	gIsFeatNo	gIsBGNon	gIsFeatPo	gIsBGPopu	IsMan
11	DATA	1	1	1	260	1	GE_Bright	GE_Bright	16543.9	273.79	1.14E+04	1.14E+03	10506	41	1.19E+01	0	0	0	0	0	
12	DATA	2	1	2	66	1	DarkCorne	DarkCorne	16564.1	273.356	4.96E+00	4.99E+00	113	41	1.22E+01	0	0	0	0	0	
13	DATA	3	1	3	66	1	DarkCorne	DarkCorne	16586.2	273.462	4.93E+00	4.97E+00	113	42	1.23E+01	0	0	0	0	0	
14	DATA	4	1	4	0	0	A_51_P39	NM_01574	16607.4	273.62	3.47E+03	3.47E+02	3322	43	1.24E+01	0	0	0	0	0	
15	DATA	5	1	5	0	0	A_55_P25	NR_02837	16628.6	273.414	1.89E+01	5.26E+00	126	43	1.24E+01	0	0	0	0	0	
16	DATA	6	1	6	0	0	A_55_P28	NM_02568	16649.9	273.518	1.32E+01	5.06E+00	118	42	1.20E+01	0	0	0	0	0	
17	DATA	7	1	7	0	0	A_55_P24	AK084197	16670.6	273.393	1.06E+02	1.17E+01	199	42	1.19E+01	0	0	0	0	0	
18	DATA	8	1	8	0	0	A_55_P27	AK184380	16692	273.5	6.52E+02	6.54E+01	718	42	1.18E+01	0	0	0	0	0	
19	DATA	9	1	9	0	0	A_51_P21	NM_01374	16713	273.5	6.58E+02	6.60E+01	720.5	41	1.18E+01	0	0	0	0	0	
20	DATA	10	1	10	0	0	A_66_P12	AK182395	16733.8	273.453	1.61E+01	5.07E+00	121	41	1.18E+01	0	0	0	0	0	
21	DATA	11	1	11	0	0	A_51_P22	NM_15351	16755.1	273.403	4.36E+02	4.39E+01	519	41	1.18E+01	0	0	0	0	0	
22	DATA	12	1	12	0	0	A_55_P28	XR_87721	16776.3	273.481	2.18E+01	5.25E+00	128	41	1.18E+01	0	0	0	0	0	
23	DATA	13	1	13	0	0	A_55_P27	NM_00115	16797.6	273.5	1.28E+04	1.28E+03	12054	40	1.17E+01	0	0	0	0	0	
24	DATA	14	1	14	0	0	A_55_P27	NM_02665	16818.7	273.5	6.04E+02	6.06E+01	666	41	1.19E+01	0	0	0	0	0	
25	DATA	15	1	15	0	0	A_55_P21	NM_00108	16839.9	273.603	1.79E+02	1.85E+01	279	41	1.18E+01	0	0	0	0	0	
26	DATA	16	1	16	0	0	A_52_P11	AK082550	16860.9	273.597	6.15E+01	7.75E+00	158	41	1.17E+01	0	0	0	0	0	
27	DATA	17	1	17	0	0	A_66_P13	NM_00790	16882.5	273.731	1.71E+04	1.71E+03	16475	41	1.14E+01	0	0	0	0	0	
28	DATA	18	1	18	0	0	A_55_P28	NM_00820	16903	273.5	4.50E+02	4.52E+01	521.5	41	1.16E+01	0	0	0	0	0	
29	DATA	19	1	19	0	0	A_55_P27	ENSMUST	16924.7	273.618	6.61E+01	8.10E+00	169	41	1.18E+01	0	0	0	0	0	



[Figure-3 Spliced Read alignment]



[Figure -4 Read Alignment]



[Figure-5 Volcano visuz Diagram]

- ***Input files***: Raw sequence reads in *fastq* formats. We had included 12 *fastq* files (two files each for six samples) in the supplementary materials for this example. Each of these files contains about five million reads of 90 base-pair (bp) which should be originated from the chromosome 19 of mouse genome.

  Step 1. Quality check on the raw reads.

  Create a directory named FastQC to store the results later. Then, call FastQC to obtain quality check metrics to inspect the quality of raw sequence reads (stored in the Fastq directory) and output the metrics to FastQC directory:

  $ mkdir FastQC
  $ fastqc Fastq/*.fastq –o FastQC

  FastQC provides a quick view on the quality of the raw sequence reads from multiple analyses, ranging from the sequence quality, GC content, to library complexity. The command above will produce a report in HTML format which could be viewed from web browser. In the report, each metric evaluated will be annotated with a green check, Red Cross, or yellow exclamation mark to indicate pass, fail, or caution respectively. Usually, the quality score

(Fig. 6) and A,C,G,T content across bases could be used to decide how the reads should be groomed prior to mapping.

Step 2. Groom raw reads.

Remove sequences with low quality to get better alignment in the later steps. Based on the FastQC reports from above commands for this example, the qualities of the reads are fine except random distribution of sequence content at the 5' end of reads. Thus, we trim 10bp from the beginning of each read:

Step 3. Align raw reads to reference genome

Map the trimmed sequence reads to reference genome using tophat2. Specifically, we first create a directory with the sample name to store the output then calling tophat2 and output results to the directory's sub-directory named Tophat_Out. Since this is a mouse sample, UCSC mm10 is used as the reference genome and the reference transcriptome file and bowtie index files are stored under Indexes directory in the home directory. Bowtie2 index files contain the genome sequences to be aligned to in bowtie2 format. Tophat2 uses bowtie2 as the base sequence aligner.

Step 4. Assemble gene expression from aligned reads

Use HTSeq to quantify the number of reads mapped to each gene:

```
$ samtools view MT1/Tophat_Out/accepted_hits.sorted.bam | python -m
HTSeq.scripts.count -q -s no - ~/Indexes/Mus_musculus/UCSC/mm10/Genes/genes.gtf >
MT1/MT1.count.txt
```

Repeat this command to the other five samples and change the input and output file names accordingly as in last step. Upon successful execution of this step, six text files suffixed with "count.txt" will be generated with each prefixed with the sample ID and stored under the subdirectory of the sample, for example MT1/MT1.count.txt, MT2/MT2.count.txt, and so son.

Step 5. Differential gene expression analysis from assembled gene expression

Take the steps as suggested by DESeq2 manual to carry out the analysis. These standard steps are listed in the following five sub-steps:

5.1. Launch RStudio and load necessary library

```
> library("DESeq2")
```

5.2. Create necessary data object.

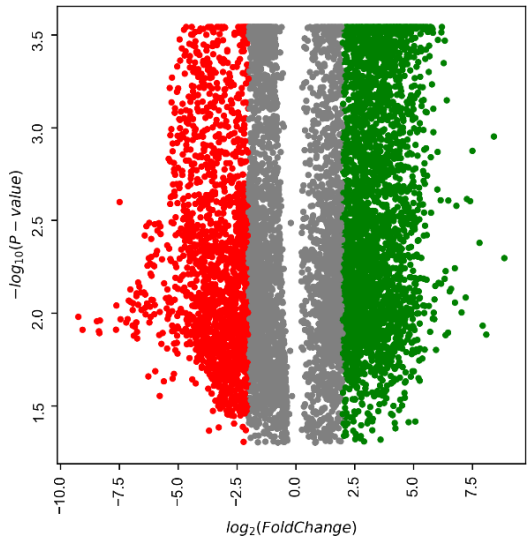
```
> sample.names <- sort(paste(c("MT", "WT"), rep(1:3, each=2), sep=""))
> file.names <- paste("../", sample.names, "/", sample.names, ".count.txt", sep="")
> conditions <- factor(c(rep("MT", 3), rep("WT", 3)))
> sampleTable <- data.frame(sampleName=sample.names,
fileName=file.names,
condition=conditions)
> # read in the HTSeq count data
> ddsHTSeq<-DESeqDataSetFromHTSeqCount(sampleTable=sampleTable, directory=".",
design=~ condition )
```

In this sub-step, we specify the sample identifiers, name of files with gene counts for each sample, and experiment condition(s) for each sample; then pass this information to DESeqDataSetFromHTSeqCount function to make a DESeqDataSet object for following analysis.

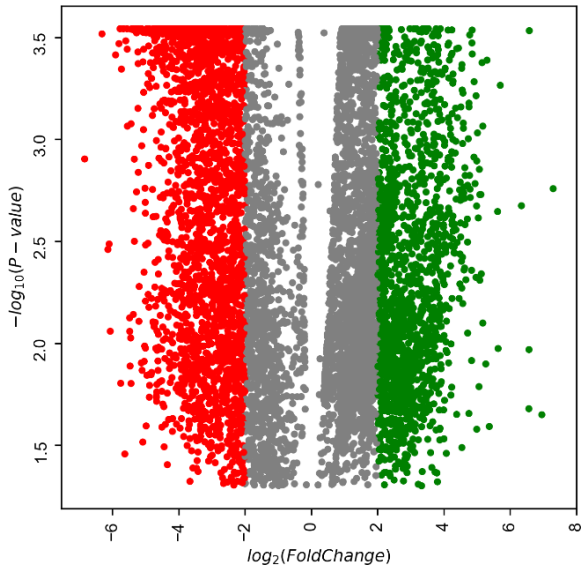
Draw PCA plot (Fig. 3) and correlation heatmap (Fig. 6) to visualize if the samples cluster per their conditions. In this example, samples are clustered into two groups, which are their genotypes. In cases where samples do cluster in groups but the grouping is not the experimental conditions or genotypes, it indicates that the samples are clustered by other factors. These factors could be latent biological subtypes or technical factors such as the batch effect. For example, the samples in supplemental Figure 6 are prepared in two different batches and PCA analysis shows two clusters corresponding to the batches, instead of four designed phenotype groups. Call the results function from DESeq2 package to extract the results from the differential gene analysis. These results include base means across samples, log2 fold changes, standard errors, test

statistics, p-values and adjusted p-values. Then, combine the results obtained with the average expressions for each genotype (grp.mean) and normalized read counts (norm.counts) for each sample into a data table. Finally, save the data table into a tab-delimited file. Group means and normalized read counts are useful when users want to inspect how a gene is expressed in the experiment. Furthermore, users could get these values and apply to other software for more analysis.

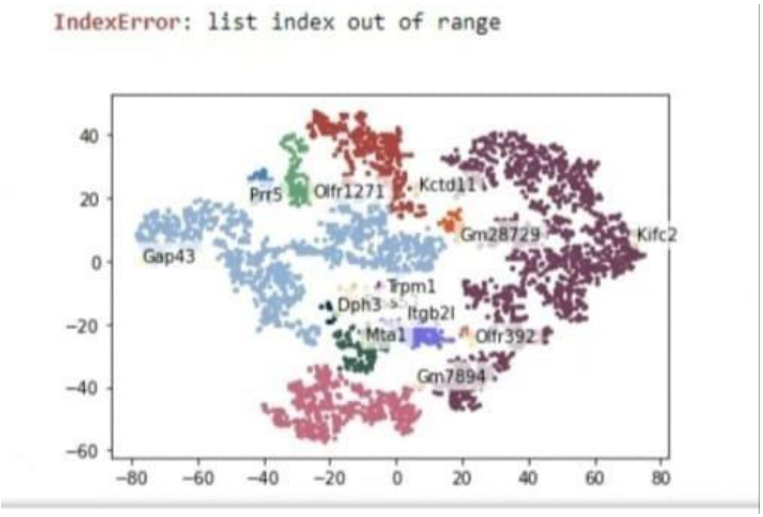
Two common plots to visualize findings from a differential gene analysis are MA plot and gene expression heatmap. The MA plot (Fig. 4) shows how gene expressed between two genotypes (log fold-change in Y-axis) with respect to overall expression on all samples (in X-axis), highlights the significantly differentially expressed genes (DEGs, adjusted P-value < 0.01) in red, and annotates the five most statistical significant DEGs.



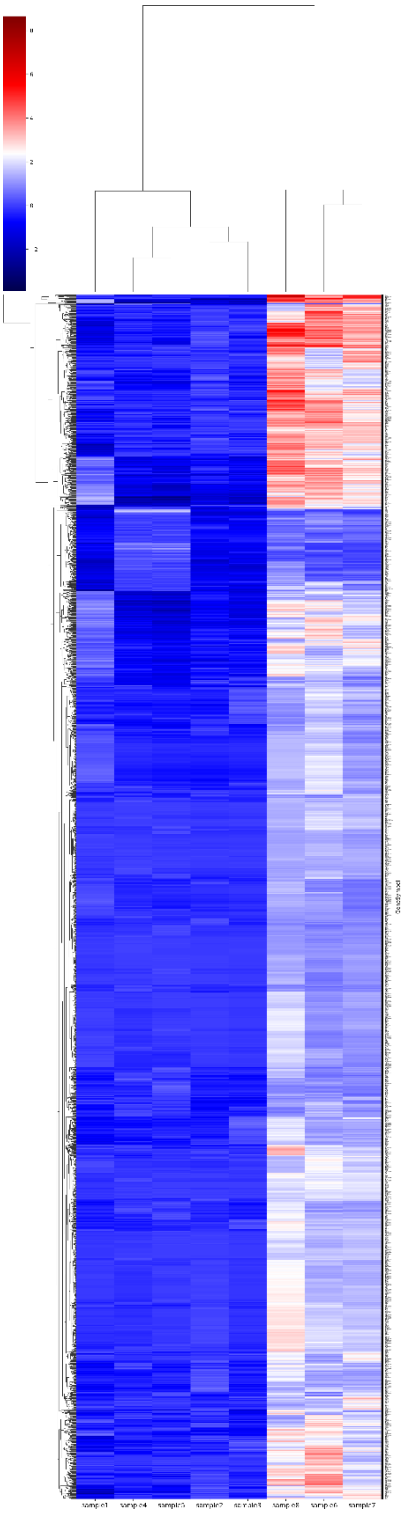
[Volcano visuzDVsDT]



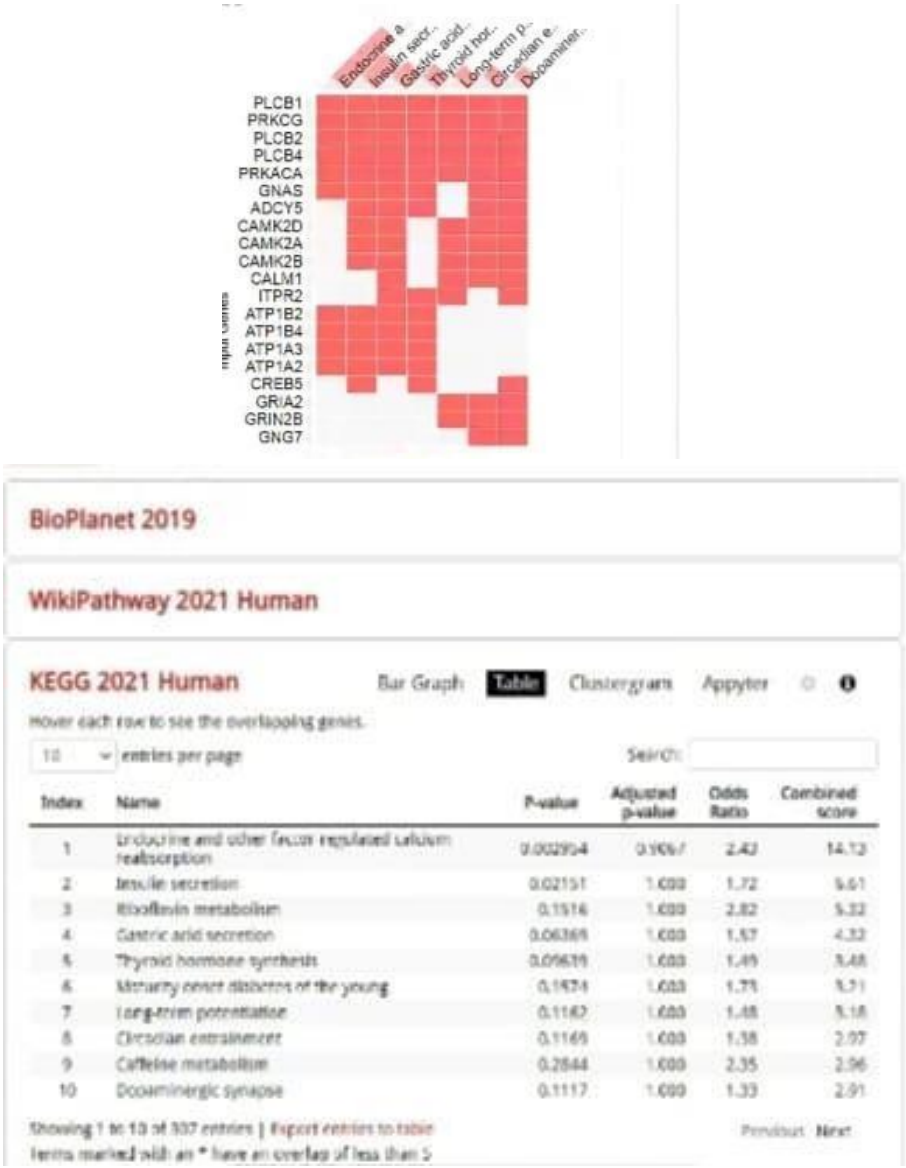
[Volcano visuzCvsD]



[Figure-5 Network Diagram]



[Figure -6 Heat map Diagram]



[Figure-7 KEGG Tabular Table]

Result Discussion:

RNA sequencing (RNA-seq) provides comprehensive insights on cellular processes (such as identifying genes that are upregulated or downregulated, etc.). However, traditional bulk RNA-seq is limited to revealing the average expression from a collection of cells, and not disambiguation single-cell behavior. Thus, it is difficult to delineate cellular heterogeneity with traditional RNA-seq, which is a disadvantage since cellular heterogeneity has been shown to play a crucial role in understanding many diseases. Therefore, researchers have turned to RNA-seq (RNAseq) in order to identify cellular heterogeneity within tissues.

RNAseq technologies have been instrumental in the study of key biological processes in many diseases, such as cancer, Alzheimer's cardiovascular diseases etcetera RNA sequencing of cells at a single-cell resolution, RNAseq, generally consists of four stages: Due to the unbiased capture of mRNA, NGS-based technologies can shed light on the known and unknown morphological features using only the molecular characterization of tissues. This unbiased and untargeted nature of NGS technologies makes them ideal for studying and exploring new systems, a major advantage compared to most image-based technologies which require target genes *a priori*.

While NGS-based approaches differ in the specifics of the protocols, they all build on the idea of adding spatial barcodes before library preparation, which are then used to map transcripts back to the appropriate positions (known as spots or voxels). An example workflow of NGS-based spatial sequencing is depicted. In the following subsections, we provide a general overview of the four most common spatial transcriptomics technologies. Their innovation was to add spatial barcodes prior to library preparation, enabling the mapping of expressions to appropriate spatial spots.

More specifically, Ståhl *et al*. positioned oligo(dT) probes and unique spatial barcodes as microarrays of spots on the surface of slides. Next, fresh frozen tissue slices were placed on the microarray and processed to release mRNA (using enzymatic permeabilization), which then hybridized with the probes on the surface of the slides. This approach consists of (i) collecting histological imaging (using standard fixation and staining techniques, including hematoxylin and eosin (HE) staining) for investigating morphological characteristics and (ii) sequencing spatially barcoded RNA to profile gene expressions. In the initial experiments, each slide consisted of approximately 1000 spots, each of diameter 100 μm with 200 μm center-to-center distance. After the initial success of *Spatial Transcriptomics*, 10x Genomics subsequently improved the resolution (shrinking the spot diameters to 55 μm with 100 μm centre-to-centre distance) and

sensitivity (capturing more than 10^4 transcripts per spot) of the approach, and eventually commercializing it as Visium). The development and commercialization of the spatial transcriptomics resulted in relatively rapid adoption across fields, such as cancer biology, developmental biology, and neuroscience.

The histological imaging and gene expression profiling of Visium are similar to the initial approach: the staining and imaging of the tissues are through traditional staining techniques, including HE staining for visualizing tissue sections using a brightfield microscope and immunofluorescence staining to visualize protein detection in tissue sections through a fluorescent microscope.

Vi-sium protocol allows for both fresh frozen (FF) and Formalin-Fixed Paraffin-Embedded (FFPE) tissues. For FF tissues, similar to Ståhl, the tissue is permeabilized, allowing the release of mRNA, which hybridizes to the spatially barcoded oligonucleotides present on the spots. The captured mRNA then goes through a reverse transcription process that results in cRNA, which are then barcoded and pooled for generating a library.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

This thesis report discusses about the most suitable deep-learning models for currently for RNA-seq data, cell type annotation of cell clusters after unsupervised clustering is mainly conducted manually. The limitation of the manual procedure makes it impossible to generate high-quality, reproducible, and standardized annotation results for the growing number of RNA-seq datasets. An experiment has been carried out to evaluate the classification performance of this deep-learning algorithm. After the preparation of dataset, the algorithm has been trained on the train-dataset. The trained model has been tested RNA sequence data set. In cell type annotation, it is usually hard to find high-quality marker genes to describe a cell cluster. A strategy is to use genes specifically expressed in a cell cluster to mark the cell type. However, using a few marker genes is often not sufficient to distinguish a cell cluster from the others. In addition, using the whole expressed gene sets may decrease the power to find the true patterns within each cell cluster. I hope this post was able to give you an insight into various deep learning based clustering techniques. The deep learning based methods have outperformed traditional clustering techniques in many benchmarks. Most of the methods discussed are promising and there is huge potential for improvement in several datasets. If you have any questions or want to suggest any changes feel free to contact me or write a comment below.

7.2 Future Work

In the future, we intend to extend this work by

(i) Alleviating more samples by combining genomics data from different sources and training a multimodal architecture,

(ii) Comparing studies on clustering based on feature extracted by RNA vs. PCA.

(iii) Improving the explanations about the predictions using both ante-hoc and post-hoc approaches. In particular, we plan to employ multimodality since multiple factors are involved in disease diagnosis (e.g. estrogen, progesterone and epidermal growth receptors in breast cancer), AI-based diagnoses might not be trustworthy solely based on a single modality, which demands the requirements of multimodal features (e.g. RNA methylation, GE, miRNA expression and CNVs data) with a reversed time attention model and Bayesian deep learning .

References

1. Hoang T, Yin C, Zheng H, Yu C, He RL, Yau SS. A new method to cluster DNA sequences using Fourier power spectrum. Journal of Theoretical Biology. 2015;372:135–145. pmid:25747773.
2. Aleb N, Labidi N. An improved *K*-means algorithm for DNA sequence clustering. In: International Workshop on Database and Expert Systems Applications (DEXA); 2015; p. 39–42.
3. Zielezinski A, Girgis HZ, Bernard G, Leimeister CA, Tang K, Dencker T, et al. Benchmarking of alignment-free sequence comparison methods. Genome Biology. 2019;20(1):1–18. pmid:31345254.
4. Randhawa GS, Hill KA, Kari L. ML-DSP: Machine Learning with Digital Signal Processing for ultrafast, accurate, and scalable genome classification at all taxonomic levels. BMC Genomics. 2019;20(1):1–21. pmid:30943897.
5. Adetiba E, Olugbara OO. Classification of eukaryotic organisms through cepstral analysis of mitochondrial RNA. In: International Conference on Image and Signal Processing; 2016; p. 243–252.
6. Goodfellow I, Bengio Y, Courville A, Deep Learning. vol. 1. MIT press Cambridge; 2016.
7. Caron M, Bojanowski P, Joulin A, Douze M. Deep clustering for unsupervised learning of visual features. In: Proceeding of European Conference on Computer Vision; 2018; p. 132–149.
8. Ji X, Henriques JF, Vedaldi A. Invariant information clustering for unsupervised image classification and segmentation. In: Proceedings of the International Conference on Computer Vision; 2019; p. 9865–9874.
9. Oehler KL, Gray RM. Combining image classification and image compression using vector quantization. In: Proceedings of the DCC '93: Data Compression Conference; 1993; p. 2–11.
10. Allio R, Donega S, Galtier N, Nabholz B. Large variation in the ratio of mitochondrial to nuclear mutation rate across animals: Implications for genetic diversity and the use of mitochondrial DNA as a molecular marker. Molecular Biology and Evolution. 2017;34(11):2762–2772. pmid:28981721.

11. Yang Y., Xu D., Nie F., Yan S. and Zhuang Y. Image clustering using local discriminant models and global integration. IEEE Transactions on Image Processing, 2010;19(10):2761–2773. pmid:20423802.
12. Hayer J, Jadeau F, Deléage G, Kay A, Zoulim F, Combet C. HBVdb: a knowledge database for Hepatitis B Virus. Nucleic Acids Research. 2013;41(D1):D566–D570. pmid:23125365.
13. Liang Q. DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR Genomics Bioinf.* 2020;2:lqaa009.
14. Rojas-Carulla M. Genet: deep representations for metagenomics. *arXiv.* 2019;arXiv:1901:11015.
15. Deng Y. Massive single-cell RNA-seq analysis and imputation via deep learning. *bioRxiv.* 2018;2018:315556. [Google Scholar].
16. Tian Q. MRCNN: a deep learning model for regression of genome-wide RNA methylation. *BMC Genomics.* 2019;20:192. [PMC free article] [PubMed] [Google Scholar].
17. Schmidt L. Graphical workflow system for modification calling by machine learning of reverse transcription signatures. *Front. Genetics.* 2019;10:876. [PMC free article] [PubMed] [Google Scholar].
18. Abadi M. Tensorflow: large-scale machine learning on heterogeneous distributed systems. *arXiv.* 2016;arXiv:1603:04467. [Google Scholar].
19. Seide F., Agarwal A. CNTK: Microsoft's open-source deep-learning toolkit. In: Wallach H., editor. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Curran Associates, Inc.; 2016. p. 2135. [Google Scholar].
20. Torshizi A.D., Wang K. Next-generation sequencing in drug development: target identification and genetically stratified clinical trials. *Drug Discovery Today.* 2018;23:1776–1783. [PubMed] [Google Scholar].
21. Raedler L.A. Keytruda (pembrolizumab): first PD-1 inhibitor approved for previously treated unresectable or metastatic melanoma. *Am. Health Drug Benefits.* 2015;8:96. [PMC free article] [PubMed] [Google Scholar].
22. Zheng, G. X. et al. Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* **8**, 14049 (2017).
23. Zeisel, A. et al. Brain structure. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* **347**, 1138–1142 (2015).

24. Wang, B., Zhu, J., Pierson, E., Ramazzotti, D. & Batzoglou, S. Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nat. Methods* **14**, 414–416 (2017).
25. Lin, P., Troup, M. & Ho, J. W. CIDR: ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biol.* **18**, 59 (2017).
26. Arisdakessian, C., Poirion, O., Yunits, B., Zhu, X. & Garmire, L. DeepImpute: an accurate, fast and scalable deep neural network method to impute single-cell RNA-seq data.
27. Chen, J. et al. An omnibus test for differential distribution analysis of microbiome sequencing data. *Bioinformatics* **34**, 643–651 (2018).
28. Guo, X., Gao, L., Liu, X. & Yin, J. Improved deep embedded clustering with local structure preservation. In *Proc. 26th International Joint Conference on Artificial Intelligence* 1753–1759 (2017).
29. incent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proc. 25th International Conference on Machine Learning* 1096–1103 (2008).
30. Zappia, L., Phipson, B. & Oshlack, A. Splatter: simulation of single-cell RNA sequencing data. *Genome Biol.* **18**, 174 (2017).
31. Dizaji, K. G., Herandi, A., Deng, C., Cai, W. & Huang, H. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proc. IEEE International Conference on Computer Vision* 5747–5756 (IEEE, 2017).
32. Abadi, M. et al. TensorFlow: a system for large-scale machine learning. In *Proc. 12th USENIX Conference on Operating Systems Design and Implementation* 265–283 (USENIX Association, 2016).
33. Kingma, D. P. & Welling, M. Stochastic gradient VB and the variational auto-encoder. In *Second International Conference on Learning Representations* (2014).
34. Wood D.E., Salzberg S.L. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* 2014;15:1–12. [PMC free article] [PubMed] [Google Scholar].
35. Busia A. A deep learning approach to pattern recognition for short RNA sequences. *bioRxiv.* 2019;2019:353474. [Google Scholar].
36. Deng Y. Massive RNA-seq analysis and imputation via deep learning. *bioRxiv.* 2018;2018:315556. [Google Scholar].

37. Tian T. Clustering single-cell RNA-seq data with a model-based deep learning approach. *Nat. Mach. Intell.* 2019;1:191–198. [Google Scholar].
38. Zhang S.Y. FunDMDeep-m6A: identification and prioritization of functional differential m6A methylation genes. *Bioinformatics.* 2019;35:i90–i98. [PMC free article] [PubMed] [Google Scholar].
39. Quang D., Xie X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of RNA sequences. *Nucleic Acids Res.* 2016;44:e107. [PMC free article] [PubMed] [Google Scholar].
40. Chollet F. Astrophysics Source Code Library; 2018. Keras: The Python Deep Learning Library. [Google Scholar].
41. Avsec Ž. Base-resolution models of transcription factor binding reveal soft motif syntax. *bioRxiv.* 2020;2020:737981. [Google Scholar].
42. Khatoon Z. Introduction to RNA-Seq and its applications to drug discovery and development. *Drug Dev. Res.* 2014;75:324–330. [PubMed] [Google Scholar].
43. Raedler L.A. Keytruda (pembrolizumab): first PD-1 inhibitor approved for previously treated unresectable or metastatic melanoma. *Am. Health Drug Benefits.* 2015;8:96. [PMC free article] [PubMed] [Google Scholar].
44. Harper A.R., Topol E.J. Pharmacogenomics in clinical practice and drug development. *Nat. Biotechnol.* 2012;30:1117–1124. [PMC free article] [PubMed] [Google Scholar].